

--	--	--	--	--	--	--	--	--	--

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

ECP2036 – MICROPROCESSOR SYSTEMS AND INTERFACING (ME)

22 OCTOBER 2016
2.30 P.M. – 4.30 P.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 7 pages with 4 questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided.
4. Opcode map and Special Function Register formats are provided in Appendices.

Question 1

- (a) What do these acronyms stand for? Briefly explain the functional description for each of them.
- (i) PC [2 marks]
 - (ii) ALU [2 marks]
- (b) Answer the following questions:
- (i) Given a 64-bit laptop computer with a memory capacity of 4Gbytes. Determine the number of data and address bus of the system? State your work and explanation. [3 marks]
 - (ii) A memory block has 8 address lines and 16 data lines. Specify its capacity. [2 marks]
- (c) An 8051 microcontroller based system is to be designed with requirement of external 32Kbytes ROM and 32Kbytes RAM, with start-up code stored in ROM. Given an available 8Kbytes program memory and 16Kbytes data memory blocks, answer the following questions:
- (i) What is the size of the data bus of this system? [1 mark]
 - (ii) Identify the number of ROM and RAM blocks needed for the system. [2 marks]
 - (iii) Calculate the address lines needed by each of these ROM and RAM blocks. [3 marks]
 - (iv) With ROM occupying the first 32Kbytes of memory space, determine the starting and ending addresses for each memory blocks. [3 marks]
 - (v) Draw the configuration of this system showing the 8051 signal lines to be used for address, data and control buses. (*Hint: You may use decoder where necessary*) [7 marks]

Question 2

- (a) What is Opcode and Operand? [2 Marks]
- (b) What are the states of the carry flag, the auxiliary carry flag, the overflow flag, the parity bit and the content of the accumulator after execution of the following instruction sequence?
- ```
MOV R2, #8
MOV A, #2
ADD A, R2
```
- [6 marks]
- (c) Write an instruction sequence to perform the following task:
- (i) Set bit addresses 69H, 6AH and 6DH. [2 marks]

**Continued...**

- (ii) Read bit 0 and bit 1 of Port 1 and write a status condition to bit 6 of Port 2 as follows: If either bit read is 1, write a 0 to the output status bit, otherwise write a 1. [3 marks]
- (d) The following is an 8051 instruction:  
***MOV 50H, #0FFH***
- (i) What is the opcode for this instruction? [1 mark]
- (ii) What are the machine language bytes for this instructions? Explain the purpose of each byte of this instruction. [3 marks]
- (iii) If an 8051 is operating from a 16 MHz crystal, how long does this instruction take to execute? [3 marks]
- (e) Write an 8051 assembly instruction sequence to add the data stored in ROM at address 0F01H to the data stored in internal RAM location pointed by R0 and store the result in register R1. [5 Marks]

### **Question 3**

- (a) Write an 8051 assembly language programme to generate a 50Hz square pulse on port pin P1.0 using Timer 0. Explain your timer setting in details and assume a 12MHz crystal. [13 marks]
- (b) Write the assembly language programme to transmit characters "8051" continuously using 8-bit UART serial protocol with 9,600 baud rate. In your programme, you must include the function TRANSMIT so that you can call it repeatedly. (Assume SMOD = 0 and 11.0592MHz crystal is used). [12 marks]

### **Question 4**

- (a) Compare the differences between interrupts and polling methods. [6 marks]
- (b) Two dual carriage roads meet at an intersection, as shown in *Figure 1* below. Four traffic lights (E, S, W and N) placed at this junction are to be controlled by an 8051 microcontroller. Each traffic light is controlled via different I/O pins on the microcontroller, as shown in *Table 1*. The operations of the traffic lights are summarized in *Table 2*. The operation cycle repeats indefinitely until the microcontroller is powered down. Assume that the traffic lights are properly connected to the microcontroller via an interfacing circuit, write a program for the 8051 microcontroller to control these traffic lights. [19 marks]

*Table 1:*

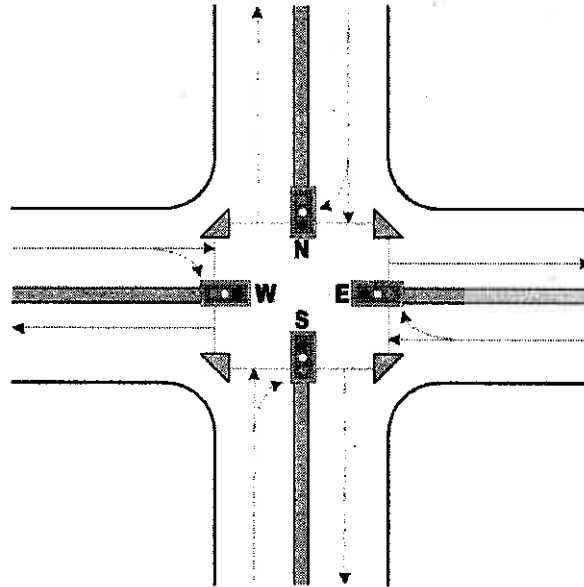
|       | Port 1 |     |     |     |     |     | Port 2 |     |     |     |     |     |
|-------|--------|-----|-----|-----|-----|-----|--------|-----|-----|-----|-----|-----|
| Pin   | 0      | 1   | 2   | 3   | 4   | 5   | 0      | 1   | 2   | 3   | 4   | 5   |
| Light | E-R    | E-Y | E-G | W-R | W-Y | W-G | N-R    | N-Y | N-G | S-R | S-Y | S-G |

Note: R: Red, Y: Yellow, G: Green; E: East, W: West, N: North, S: South

**Continued ...**

*Table 2:*

| Duration | E      | W      | N      | S      |
|----------|--------|--------|--------|--------|
| 60s      | Red    | Red    | Red    | Green  |
| 3s       | Red    | Red    | Red    | Yellow |
| 30s      | Red    | Red    | Green  | Red    |
| 3s       | Red    | Red    | Yellow | Red    |
| 60s      | Red    | Green  | Red    | Red    |
| 3s       | Red    | Yellow | Red    | Red    |
| 30s      | Green  | Red    | Red    | Red    |
| 3s       | Yellow | Red    | Red    | Red    |

*Figure 1*

Continued...

## Appendix A: Opcode Map

| Label | 0                        | 1                         | 2                      | 3                       | 4                           | 5                           | 6                           | 7                           | 8                          | 9                           | A                          | B                                 | C                       | D                          | E                         | F                       |
|-------|--------------------------|---------------------------|------------------------|-------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|----------------------------|-----------------------------|----------------------------|-----------------------------------|-------------------------|----------------------------|---------------------------|-------------------------|
| 0     | 1B, 1C<br>NOP            | 3B, 3C<br>JBC<br>b1cd     | 3B, 3C<br>JB<br>b1cd   | 2B, 2C<br>JNC<br>nd     | 2B, 2C<br>JC<br>rel         | 2B, 2C<br>JNC<br>nd         | 2B, 2C<br>JZ<br>rel         | 3B, 2C<br>JNZ<br>rel        | 3B, 2C<br>SIMP<br>rel      | 3B, 2C<br>MOV<br>DPR, #16   | 3B, 2C<br>ORL<br>C, #bit   | 2B, 2C<br>ANL<br>C, #bit          | 2B, 2C<br>PUSH<br>dir   | 2B, 2C<br>POP<br>dir       | 1B, 2C<br>MOVX<br>A, #DPR | 1B, 2C<br>MOVX<br>@DPR  |
| 1     | 2B, 2C<br>AJMP<br>(P0)   | 2B, 2C<br>ACALL<br>(P0)   | 2B, 2C<br>AJMP<br>(P1) | 2B, 2C<br>ACALL<br>(P1) | 2B, 2C<br>AJMP<br>(P2)      | 2B, 2C<br>ACALL<br>(P2)     | 2B, 2C<br>AJMP<br>(P3)      | 2B, 2C<br>ACALL<br>(P3)     | 2B, 2C<br>AJMP<br>(P4)     | 2B, 2C<br>ACALL<br>(P4)     | 2B, 2C<br>AJMP<br>(P5)     | 2B, 2C<br>ACALL<br>(P5)           | 2B, 2C<br>AJMP<br>(P6)  | 2B, 2C<br>ACALL<br>(P6)    | 2B, 2C<br>AJMP<br>(P7)    | 2B, 2C<br>ACALL<br>(P7) |
| 2     | 3B, 2C<br>LJMP<br>addr16 | 3B, 2C<br>LCALL<br>addr16 | 1B, 1C<br>RET          | 1B, 1C<br>RETI          | 2B, 1C<br>ORL<br>dir, A     | 2B, 1C<br>ANL<br>dir, A     | 2B, 1C<br>XRL<br>dir, A     | 2B, 2C<br>ORL<br>C, bit     | 2B, 2C<br>ANL<br>C, bit    | 2B, 2C<br>MOV<br>bit, C     | 2B, 1C<br>MOV<br>C, bit    | 2B, 1C<br>CPL<br>bit              | 2B, 1C<br>CLR<br>bit    | 2B, 1C<br>SETB<br>bit      | 1B, 2C<br>MOVX<br>A, #R0  | 1B, 2C<br>MOVX<br>@R0   |
| 3     | 1B, 1C<br>RR             | 1B, 1C<br>RRC             | 1B, 1C<br>RL           | 1B, 1C<br>RLC           | 3B, 2C<br>ORL<br>dir, #data | 3B, 2C<br>ANL<br>dir, #data | 3B, 2C<br>XRL<br>dir, #data | 1B, 2C<br>JMP<br>#data      | 1B, 2C<br>MOVC<br>A, #A+PC | 1B, 2C<br>MOVC<br>A, #A+DPR | 1B, 2C<br>INC<br>DPR       | 2B, 1C<br>CPL<br>C                | 2B, 1C<br>CLR<br>C      | 2B, 1C<br>SETB<br>C        | 1B, 2C<br>MOVX<br>@R1     | 1B, 2C<br>MOVX<br>@R1A  |
| 4     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, #data   | 2B, 1C<br>ANL<br>A, #data   | 2B, 1C<br>XRL<br>A, #data   | 2B, 1C<br>MOV<br>A, #data   | 1B, 2C<br>DIV<br>AB        | 2B, 1C<br>SUBB<br>A, #data  | 1B, 2C<br>MUL<br>AB        | 3B, 2C<br>CJNE<br>A, #data, rel   | 1B, 1C<br>SWAP<br>A     | 1B, 1C<br>DA               | 1B, 1C<br>CLR             | 1B, 1C<br>CPL           |
| 5     | 2B, 1C<br>INC            | 2B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, dir     | 2B, 1C<br>ANL<br>A, dir     | 2B, 1C<br>XRL<br>A, dir     | 2B, 2C<br>MOV<br>dir, #data | 2B, 2C<br>MOV<br>dir, dir  | 2B, 2C<br>SUBB<br>A, dir    | 2B, 2C<br>MOVB<br>A, dir   | 3B, 2C<br>CJNE<br>A, dir, rel     | 1B, 1C<br>XCH<br>A, dir | 1B, 1C<br>DINZ<br>dir, rel | 1B, 1C<br>MOV<br>A, dir   | 1B, 1C<br>MOV<br>dir, A |
| 6     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, #R0     | 2B, 1C<br>ANL<br>A, #R0     | 2B, 1C<br>XRL<br>A, #R0     | 2B, 1C<br>MOV<br>@R0, #data | 2B, 2C<br>MOVB<br>A, #R0   | 2B, 2C<br>SUBB<br>A, #R0    | 2B, 2C<br>MOVB<br>@R0, dir | 3B, 2C<br>CJNE<br>@R0, #data, rel | 1B, 1C<br>XCH<br>A, #R0 | 1B, 1C<br>XCHD<br>R0, rel  | 1B, 1C<br>MOV<br>@R0      | 1B, 1C<br>MOV<br>@R0A   |
| 7     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, #R1     | 2B, 1C<br>ANL<br>A, #R1     | 2B, 1C<br>XRL<br>A, #R1     | 2B, 1C<br>MOV<br>@R1, #data | 2B, 2C<br>MOVB<br>dir, #R1 | 2B, 2C<br>SUBB<br>A, #R1    | 2B, 2C<br>MOVB<br>@R1, dir | 3B, 2C<br>CJNE<br>@R1, #data, rel | 1B, 1C<br>XCH<br>A, #R1 | 1B, 1C<br>XCHD<br>A, #R1   | 1B, 1C<br>MOV<br>A, #R1   | 1B, 1C<br>MOV<br>@R1A   |
| 8     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R0      | 2B, 1C<br>ANL<br>A, R0      | 2B, 1C<br>XRL<br>A, R0      | 2B, 1C<br>MOV<br>R0, #data  | 2B, 2C<br>MOVB<br>dir, R0  | 2B, 2C<br>SUBB<br>A, R0     | 2B, 2C<br>MOVB<br>R0, dir  | 3B, 2C<br>CJNE<br>R0, #data, rel  | 1B, 1C<br>XCH<br>A, R0  | 1B, 1C<br>DINZ<br>R0, rel  | 1B, 1C<br>MOV<br>A, R0    | 1B, 1C<br>MOV<br>R0A    |
| 9     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R1      | 2B, 1C<br>ANL<br>A, R1      | 2B, 1C<br>XRL<br>A, R1      | 2B, 1C<br>MOV<br>R1, #data  | 2B, 2C<br>MOVB<br>dir, R1  | 2B, 2C<br>SUBB<br>A, R1     | 2B, 2C<br>MOVB<br>R1, dir  | 3B, 2C<br>CJNE<br>R1, #data, rel  | 1B, 1C<br>XCH<br>A, R1  | 1B, 1C<br>DINZ<br>R1, rel  | 1B, 1C<br>MOV<br>A, R1    | 1B, 1C<br>MOV<br>R1A    |
| A     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R2      | 2B, 1C<br>ANL<br>A, R2      | 2B, 1C<br>XRL<br>A, R2      | 2B, 1C<br>MOV<br>R2, #data  | 2B, 2C<br>MOVB<br>dir, R2  | 2B, 2C<br>SUBB<br>A, R2     | 2B, 2C<br>MOVB<br>R2, dir  | 3B, 2C<br>CJNE<br>R2, #data, rel  | 1B, 1C<br>XCH<br>A, R2  | 1B, 1C<br>DINZ<br>R2, rel  | 1B, 1C<br>MOV<br>A, R2    | 1B, 1C<br>MOV<br>R2A    |
| B     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R3      | 2B, 1C<br>ANL<br>A, R3      | 2B, 1C<br>XRL<br>A, R3      | 2B, 1C<br>MOV<br>R3, #data  | 2B, 2C<br>MOVB<br>dir, R3  | 2B, 2C<br>SUBB<br>A, R3     | 2B, 2C<br>MOVB<br>R3, dir  | 3B, 2C<br>CJNE<br>R3, #data, rel  | 1B, 1C<br>XCH<br>A, R3  | 1B, 1C<br>DINZ<br>R3, rel  | 1B, 1C<br>MOV<br>A, R3    | 1B, 1C<br>MOV<br>R3A    |
| C     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R4      | 2B, 1C<br>ANL<br>A, R4      | 2B, 1C<br>XRL<br>A, R4      | 2B, 1C<br>MOV<br>R4, #data  | 2B, 2C<br>MOVB<br>dir, R4  | 2B, 2C<br>SUBB<br>A, R4     | 2B, 2C<br>MOVB<br>R4, dir  | 3B, 2C<br>CJNE<br>R4, #data, rel  | 1B, 1C<br>XCH<br>A, R4  | 1B, 1C<br>DINZ<br>R4, rel  | 1B, 1C<br>MOV<br>A, R4    | 1B, 1C<br>MOV<br>R4A    |
| D     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R5      | 2B, 1C<br>ANL<br>A, R5      | 2B, 1C<br>XRL<br>A, R5      | 2B, 1C<br>MOV<br>R5, #data  | 2B, 2C<br>MOVB<br>dir, R5  | 2B, 2C<br>SUBB<br>A, R5     | 2B, 2C<br>MOVB<br>R5, dir  | 3B, 2C<br>CJNE<br>R5, #data, rel  | 1B, 1C<br>XCH<br>A, R5  | 1B, 1C<br>DINZ<br>R5, rel  | 1B, 1C<br>MOV<br>A, R5    | 1B, 1C<br>MOV<br>R5A    |
| E     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R6      | 2B, 1C<br>ANL<br>A, R6      | 2B, 1C<br>XRL<br>A, R6      | 2B, 1C<br>MOV<br>R6, #data  | 2B, 2C<br>MOVB<br>dir, R6  | 2B, 2C<br>SUBB<br>A, R6     | 2B, 2C<br>MOVB<br>R6, dir  | 3B, 2C<br>CJNE<br>R6, #data, rel  | 1B, 1C<br>XCH<br>A, R6  | 1B, 1C<br>DINZ<br>R6, rel  | 1B, 1C<br>MOV<br>A, R6    | 1B, 1C<br>MOV<br>R6A    |
| F     | 1B, 1C<br>INC            | 1B, 1C<br>DEC             | 2B, 1C<br>ADD          | 2B, 1C<br>ADDC          | 2B, 1C<br>ORL<br>A, R7      | 2B, 1C<br>ANL<br>A, R7      | 2B, 1C<br>XRL<br>A, R7      | 2B, 1C<br>MOV<br>R7, #data  | 2B, 2C<br>MOVB<br>dir, R7  | 2B, 2C<br>SUBB<br>A, R7     | 2B, 2C<br>MOVB<br>R7, dir  | 3B, 2C<br>CJNE<br>R7, #data, rel  | 1B, 1C<br>XCH<br>A, R7  | 1B, 1C<br>DINZ<br>R7, rel  | 1B, 1C<br>MOV<br>A, R7    | 1B, 1C<br>MOV<br>R7A    |

**Continued...**

## Appendix B: Special Function Register Format

TMOD : [Bit 0 (LSB) to Bit 3 is for Timer 0 and Bit 4 to Bit 7 (MSB) is for Timer 1]

| GATE | C / T | M1 | M0 | GATE | C / T | M0 | M1 |
|------|-------|----|----|------|-------|----|----|
|------|-------|----|----|------|-------|----|----|

GATE: Timer only runs while /INT1 is set.

C / T: '1' for event counter, '0' for interval timer

M1, M0: Mode bit select

"00" Mode 0 – 13-bit timer mode

"01" Mode 1 – 16-bit timer mode

"10" Mode 2 – 8-bit auto-reload mode

"11" Mode 3 – Split timer mode

TCON :

| TF1 | TR1 | TFO | TRO | IE1 | IT1 | IE0 | IT0 |
|-----|-----|-----|-----|-----|-----|-----|-----|
|-----|-----|-----|-----|-----|-----|-----|-----|

TCON.7 TF1 Timer 1 overflow flag. Set by hardware on overflow.

Clear by hardware when processor vectors to interrupt routine.

TCON.6 TRI Timer 1 run control bit. Set/cleared by software to start/stop timer.

TCON.5 TFO Timer 0 overflow flag. Set by hardware on overflow.

Clear by hardware when processor vectors to interrupt routine.

TCON.4 TRO Timer 0 run control bit. Set/cleared by software to start/stop timer.

TCON.3 IE1 Interrupt 1 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.

TCON.2 IT1 Interrupt 1 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.

TCON.1 IE0 Interrupt 0 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.

TCON.0 IT0 Interrupt 0 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.

SCON :

| SMO | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|-----|-----|-----|-----|-----|-----|----|----|
|-----|-----|-----|-----|-----|-----|----|----|

SMO SM1

0 0 = Shift register mode

0 1 = 8-bit UART mode

1 0 = 9-bit UART mode (Fixed Baud Rate)

1 1 = 9-bit UART mode (Variable Baud Rate)

SM2 = '1' = Enable multiprocessor communication

REN = Receiver Enable

TB8 = Transmit Bit

TI = Transmit Interrupt

RI = Receive Interrupt

Continued...

**IE:**

|    |  |     |    |     |     |     |     |
|----|--|-----|----|-----|-----|-----|-----|
| EA |  | ET2 | ES | ET1 | EX1 | ET0 | EXO |
|----|--|-----|----|-----|-----|-----|-----|

| Bit Position | Symbol | Bit Address | Description                                                                                                                                              |
|--------------|--------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| IE.7         | EA     | AFH         | Global enable/disable.<br>EA = '1', each individual source is enable/disable<br>By setting/clearing its enable bit.<br>EA = '0', disable all interrupts. |
| IE.6         | -      | AEH         | Undefined                                                                                                                                                |
| IE.5         | -      | ADH         | Not implemented in 8051. ET2 for 8052.                                                                                                                   |
| IE.4         | ES     | ACH         | Serial port interrupt enable bit.                                                                                                                        |
| IE.3         | ET1    | ABH         | Timer1 interrupt enable bit.                                                                                                                             |
| IE.2         | EX1    | AAH         | External interrupt enable bit.                                                                                                                           |
| IE.1         | ET0    | A9H         | Timer0 interrupt enable bit.                                                                                                                             |
| IE.0         | EXO    | A8H         | External interrupt enable bit.                                                                                                                           |

**IP:**

|  |  |     |    |     |     |     |     |
|--|--|-----|----|-----|-----|-----|-----|
|  |  | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|--|--|-----|----|-----|-----|-----|-----|

|      |     |     |                                        |
|------|-----|-----|----------------------------------------|
| IP.7 | -   | -   | Undefined.                             |
| IP.6 | -   | -   | Undefined.                             |
| IP.5 | -   | BDH | Not implemented in 8051. PT2 for 8052. |
| IP.4 | PS  | BCH | Serial port interrupt priority bit.    |
| IP.3 | PT1 | BBH | Timer1 interrupt priority bit.         |
| IP.2 | PX1 | BAH | External interrupt priority bit.       |
| IP.1 | PT0 | B9H | Timer-0 interrupt priority bit.        |
| IP.0 | PX0 | B8H | External interrupt priority bit.       |

**Selected Interrupt Vectors**

| Interrupt source | Flag       | Vector Address |
|------------------|------------|----------------|
| System Reset     | RST        | 0000H          |
| External 0       | IE0        | 0003H          |
| Timer 2 (8052)   | TF2 & EXF2 | 002BH          |

**PSW:**

|    |    |    |     |     |    |   |   |
|----|----|----|-----|-----|----|---|---|
| CY | AC | F0 | RS1 | RS0 | OV | - | P |
|----|----|----|-----|-----|----|---|---|

CY : Carry Flag

AC : Auxiliary Carry Flag

RS1, RS0: Register Bank Select

OV : Overflow Flag

P : Parity

**End of Paper**